Informatique et sciences du numérique
Langages et programmation

---

7. Les tableaux en Python

Lorsque l'on écrit un programme, on a souvent besoin de créer des variables qui contiennent une succession de valeurs plutôt qu'une seule valeur. Par exemple,
* la liste des 10 plus petits nombres premiers dans l'ordre croissant ;
* la liste des pays de l'union européenne ;
* une liste de 30 nombres entiers compris entre 0 et 100 choisis aléatoirement ;
etc.

Dans de nombreux langages de programmation, ce type de variable s'appelle « un tableau ». En Python, on dit qu'il s'agit d'une variable de type **list**. Nous utiliserons donc le terme « tableau » pour désigner ce type de variable en français et le terme « **list** » pour faire référence à son type en Python.

1 Créer et consulter un tableau

• Pour créer un tableau avec quelques valeurs, on écrit

nomVariable **=** **[***valeur1***,** *valeur2***,** ...**]**

Exemples :

```
>>> listeCarres=[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> pointsCard = ["nord", "ouest", "sud", "est"]
>>> type(pointsCard)
<class 'list'>
```

• Pour créer un tableau vide, on écrit :

unTab1 **= []**

ou bien

unTab2 **= list()**

• Pour afficher la totalité du tableau, il suffit de l'appliquer à la fonction **print(...)** (ou bien, si on est dans l'interpréteur, de taper son nom puis sur la touche « Entrée »)
Exemple : Dans un fichier essai.py tapez puis exécutez les deux instructions suivantes :

```
listeCarres=[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
print(listeCarres)
```

• Pour désigner un des éléments du tableau, on écrit

*unTab***[***ind***]**

où *unTab* est le nom du tableau et *ind* est l'indice de l'élément dans ce tableau. L'indice est le numéro de l'élément dans la succession des valeurs formée par le tableau.

Attention ! Le premier élément d'un tableau possède l'indice 0, le deuxième élément possède l'indice 1, etc.

Exemple :

```
>>> listeCarres[3]
16
```

• Pour connaître le nombre d'éléments dans un tableau, on utilise la fonction **len(**...**)**. Par exemple, on peut écrire l'instruction suivante :

nbElements **= len(***unTab***)**

Puisque le premier élément d'un tableau *unTab* possède l'indice 0, le dernier élément possède l'indice $\text{len}(unTab) - 1$.

Exemples :

```
>>> len(listeCarres)
10
>>> listeCarres[9]
100
>>> listeCarres[10]
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    listeCarres[10]
IndexError: list index out of range
```

2 Parcourir un tableau en partie ou en totalité

• Pour parcourir un tableau en totalité, on utilise le plus souvent une boucle « for » et les fonctions **len(...)** et **range(...)**de la façon suivante :

for *ind* **in range(len(***unTab***)) :**
    …*unTab*[*ind*]…

Exemple :

```
>>> for ind in range(len(listeCarres)) :
        print(listeCarres[ind], "est le carré de", ind+1)

1 est le carré de 1
4 est le carré de 2
9 est le carré de 3
16 est le carré de 4
25 est le carré de 5
36 est le carré de 6
49 est le carré de 7
64 est le carré de 8
81 est le carré de 9
100 est le carré de 10
```

• Pour parcourir la partie du tableau comprise entre l'élement d'indice *indMin* et l'élement d'indice *indMax* inclus, il suffit de modifier le contenu de **range(...)** de la façon suivante :

for *ind* **in range(***indMin***,** *indMax+1***) :**
    …*unTab*[*ind*]…

Exemple : Pour parcourir le tableau listeCarres entre l'élement d'indice 3 et l'élément d'indice 7, on peut écrire :

```
>>> for ind in range(3, 8) :
                print(listeCarres[ind], "est le carré de", ind+1)

16 est le carré de 4
25 est le carré de 5
36 est le carré de 6
49 est le carré de 7
64 est le carré de 8
```

• On peut aussi parcourir un tableau en totalité avec une boucle « for » sans utiliser les indices et la fonction **range(...)** de la façon suivante :

for *elem* **in** *unTab* **:**

…elem…

Exemple :

```
>>> for elem in pointsCard :
        print(elem)

nord
ouest
sud
est
```

3 Modifier un tableau

• Pour changer la valeur d'un des éléments du tableau, il suffit de le manipuler comme s'il s'agissait d'une variable isolée en utilisant une instruction d'affectation de la façon suivante :

*unTab***[***ind***] =** *nouvelleValeur*

Exemple :

```
>>> mesNombres=[7, 48, 13, 84]
>>> mesNombres[2]=62
>>> mesNombres[0] += 10
>>> mesNombres
[17, 48, 62, 84]
```

• Pour ajouter un élément à la fin du tableau *unTab* , on utilise la méthode **append(…)** associé à cet objet de type **list** en lui passant en argument la valeur à ajouter de la façon suivante :

*unTab***.append(***nouvElem***)**

Exemple :

```
>>> mesNombres.append(126)
>>> mesNombres
[17, 48, 62, 84, 126]
```

• Pour supprimer le dernier élement du tableau *unTab* et le récupérer dans un variable séparée, on utilise la méthode **pop()** qui ne comporte aucun argument de la façon suivante :

uneVar **=** *unTab***.pop()**

Exemple :

```
>>> dernierElem = mesNombres.pop()
>>> dernierElem
126
>>> mesNombres
[17, 48, 62, 84]
```

4 Opérations sur les tableaux

• Pour constituer un unique tableau à partir de deux tableaux en effectuant une « concaténation » (c'est-à-dire en les mettant « bout à bout »), on utilise le signe '+' (comme pour les chaînes de caractères) de la façon suivante :

Tab3 **=** *Tab1* **+** *Tab2*

Dans exemple, le tableau *Tab3* contiendra tous les éléments de *Tab1 puis* tous les éléments de *Tab2*.

Exemple :

```
>>> monTab=[5, 8, 11, 19]
>>> tonTab=[88, 41, 198]
>>> notreTab= tonTab + monTab
```

```
>>> notreTab
[88, 41, 198, 5, 8, 11, 19]
```

• On peut constituer un nouveau tableau *nouvTab* en reproduisant *N* fois le tableau *unTab* (où *N* est de type **int**) de la façon suivante :

nouvTab **=** *N****** *unTab*

ou bien

nouvTab **=** *unTab*N*

Exemple :

```
>>> grdTab = tonTab*5
>>> grdTab
[88, 41, 198, 88, 41, 198, 88, 41, 198, 88, 41, 198, 88, 41, 198]
```

Remarque : ceci permet notamment de constituer un tableau constitué d'un très grand nombre de fois le même nombre. Par exemple, pour constituer un tableau *tresGrandTab* qui comporte 10000 fois le nombre 0 on peut écrire

*tresGrandTab***=**[0]*10000

5. Comparons les tableaux et les chaînes de caractères

a) Les points communs

Une chaîne de caractère possède certaines propriétés en commun avec les tableaux :

• On peut déterminer le nombre de caractères dans une chaîne de caractère avec la fonction **len(...)**

Exemple :

```
>>> message = "bonjour à tous"
>>> len(message)
14
```

• On peut accéder à un caractère de la chaîne de caractères par son indice

Exemple :

```
>>> message[8]
'à'
```

• On peut parcourir tous les caractères d'une chaîne avec une boucle for sans utiliser la fonction range.

Exemple : Essayez l'instruction suivante (après avoir créé une variable 'message' de type **str**) :

```
for c in message :
    print(c)
```

b) Une grande différence

• On ne peut pas modifier le contenu d'une chaîne de caractère comme on modifie le contenu d'un tableau :

Exemple

```
>>> message[10]='v'
Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    message[10]='v'
TypeError: 'str' object does not support item assignment
```